UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/786,326 | 02/26/2004 | Jun-seo Lee | Q78241 | 2660 |

23373 7590 08/04/2009

SUGHRUE MION, PLLC
2100 PENNSYLVANIA AVENUE, N.W.
SUITE 800
WASHINGTON, DC 20037

| EXAMINER | |
|---|---|
| BELANI, KISHIN G | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2443 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 08/04/2009 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 10/786,326 | LEE, JUN-SEO | |
| | Examiner | Art Unit | |
| | KISHIN G. BELANI | 2443 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *16 April 2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-11 and 14-19* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-11 and 14-19* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

## DETAILED ACTION

This action is in response to Applicant's amendment filed on 04/16/2009. **Claims 1, 7, and 14 have been amended. New claims 18 and 19 have been added. Claims 1-11 and 14-19 are now pending** in the present application. Applicant's amendments to claims are shown in ***bold and italics*** and the examiner's response to the claim amendments is shown in **bold i**n this office action. **This Action is made FINAL**.

### *Claim Rejections - 35 USC § 103*

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

The factual inquiries set forth in *Graham* **v.** *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

**Claims 1-11 and 14-16** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Crow et al. (U.S. Patent Application Publication # 2002/0161915**

**A1)** in view of **Ganesan et al. (U.S. Patent Application Publication # 2003/0069973 A1)** and further in view of **Varma et al. (U.S. Patent Application Publication # 2004/0037302 A1)** and further in view of **Rana et al. (U.S. Patent Application Publication # 2002/0095512 A1).**

Consider **claim 1**, Crow et al. show and disclose a method for receiving a plurality of packets from a network and distributing the packets to a plurality of protocol processors (Fig.1, that shows a Border Router 16 receiving a plurality of packets from the Internet 22, and distributing the packets to a plurality of protocol processors 24; paragraph 0018 discloses the same details), comprising the steps of:

if a received packet is a fragmented packet, determining whether the received packet is a first fragment packet (Flowchart of Fig. 4, blocks 102 and 108 that test for a fragmented received packet and then check if it is the first (primary) fragment packet; paragraph 0037, lines 1-5 that disclose testing for a fragmented packet; paragraph 0038 that discloses a test for the fragmented received packet being the first such packet);

searching an index indicating one of the protocol processors (Fig. 1, Translation Table 82 with entries in it; Fig. 3 that shows an entry matching the IP and Transport Header information (shown in Fig. 2A) in the primary fragment, after searching the translation table 82 for a matching protocol processor);

entering the index into the corresponding list (Fig. 3, a group of Fragment Contexts shown below Fragment Context 92 and representing a list, for the secondary segments that were previously stored, being updated with the fragment context of the primary

fragment; Fig. 4, blocks 114-116; paragraphs 0039-0040 that describe generating a fragment-context 92 (shown in Fig. 3) for the identified translation entry and applying it to the secondary fragments).

However, Crow et al. do not disclose a method wherein if the received packet is the first fragment packet, looking up a tunnel ID of the received packet from a tunnel ID look-up table; looking-up a fragment ID of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list; and if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table***; and wherein, when the list of the fragment look-up table is not accessed during a predetermined time, the list is removed from the fragment look-up table.***

In the same field of endeavor, Ganesan et al. disclose a method for looking up a tunnel ID of the received packet from a tunnel ID look-up table (Flowchart of Fig. 11, blocks 1120 and 1122; paragraph 0177 which discloses that for received packets, based on the tunnel ID of the packet, NAT (Network Address Translation) lookups and mappings are applied, thereby disclosing looking up a tunnel ID of the received packet from a tunnel ID look-up table).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide a method for looking up a tunnel ID of the received packet from a tunnel ID look-up table, as taught by Ganesan et al., in the

method of Crow et al., so that encapsulated received packets can be securely delivered through the firewall of the receiving node.

However, Crow et al., as modified by Ganesan et al., do not specifically disclose a method wherein if the received packet is the first fragment packet, looking-up a fragment ID of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list; and if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table***; and wherein, when the list of the fragment look-up table is not accessed during a predetermined time, the list is removed from the fragment look-up table.***

In the same field of endeavor, Varma et al. disclose a method wherein if the received packet is the first fragment packet, looking-up a fragment ID of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list (Fig. 3; that shows Data Memory 210 for storing packets, Control Memory 200 for storing the fragment look-up table with head and tail pointers and other data (including packet count) for each list, and Link Memory 220 with pointers to keep track of the related packets of the list in the data memory; paragraph 0032, lines 6-13 which disclose that a determination is made as to whether the received packet is a first block of data associated with this

queue, that is if the queue was empty upon the arrival of the packet; by checking the packet count value for zero. If the packet count value is zero, the head pointer and the tail pointer are both set to the address of an allocated block in the data memory to store the received packet, as this is the only packet associated with the queue, thus disclosing a method wherein if the received packet is the first fragment packet, looking-up a fragment ID (queue id) of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide a method wherein if the received packet is the first fragment packet, looking-up a fragment ID of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list, as taught by Varma et al., in the method of Crow et al., as modified by Ganesan et al., so that the related packets received in the out of order sequence, may be organized in a single list for subsequent translation and transmission to a common destination host/port.

However, Crow et al., as modified by Ganesan et al. and Varma et al., do not specifically disclose a method wherein if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table***; and wherein, when the list of the***

***fragment look-up table is not accessed during a predetermined time, the list is removed from the fragment look-up table.*** Although Crow et al. do disclose creating a fragment context for a primary fragment and associating it with the related secondary fragments for subsequent translation and transmission of the related secondary fragments, Crow et al. do not specifically associate it with a list in the fragment look-up table.

In the same field of endeavor, Rana et al. disclose a method wherein if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table (Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a fragment from a known session (i.e. belonging to an existing link list, as discussed in paragraphs 0020-0023), a fragment id is associated with the data packet, thus disclosing a method wherein if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table)***; and wherein, when the list of the fragment look-up table is not accessed during a predetermined time, the list is removed from the fragment look-up table*** **(paragraph 0027 which discloses that fragment reassembly unit 28 (shown in Fig. 1) must also check for time out conditions, to ensure that all of the datagram fragments associated with a particular PDU are received, otherwise the full PDU cannot be reassembled and both the resources of fragment memory 30 and link list memory 24 would eventually fill with these unresolved fragments and be**

**rendered useless; further disclosing that to prevent this situation, a timeout condition is generated after a fragment or fragments have been in the fragment reassembly unit for more than a programmable, predetermined amount of time; and once the timeout condition has occurred, all of the fragments for that PDU are discarded and the links in the link list memory 24 are deallocated).**

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide a method wherein if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table, and wherein, when the list of the fragment look-up table is not accessed during a predetermined time, the list is removed from the fragment look-up table as taught by Rana et al., in the method of Crow et al., as modified by Ganesan et al. and Varma et al., so that the related packets received in the out of order sequence, may be organized in a single list for subsequent translation and transmission to a common destination host/port; and to clear out the fragment list for a message, all of whose fragments haven't been received within a predetermined time, in order to free up the resources for other messages.

Consider **claim 2**, and **as it applies to claim 1 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose a method, wherein the step of entering the index into the corresponding list of the fragment look-up table, includes newly entering the result of the looked-up fragment ID and the index into the fragment look-up table, if the list corresponding to the result of the looked-up

fragment ID does not exist in the fragment look-up table (in Varma et al. reference, paragraph 0032, lines 6-13 which disclose that a determination is made as to whether the received packet is a first block of data associated with this queue, that is if the queue was empty upon the arrival of the data; by checking the packet count value for zero. If the count value is zero, the head pointer and the tail pointer are both set to the address of an allocated block in the data memory to store the received packet, as this is the only packet associated with the queue, thereby disclosing creating a new list with the fragment id (queue id), if the list corresponding to the result of the looked-up fragment ID does not exist in the fragment look-up table); (in Crow et al. reference, Fig. 3 that shows a sample address translation entry; Flowchart of Fig. 4, blocks 110-114; paragraph 0040 that discloses the process of generating an index (fragment-context 92 shown in Fig. 3) for the identified translation entry 90, thereby disclosing generating an index); and (in Rana et al. reference, Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a new fragment, in which case a fragment id (index) is assigned to the data packet); together disclosing a method wherein the step of entering the index into the corresponding list of the fragment look-up table, includes newly entering the result of the looked-up fragment ID and the index into the fragment look-up table, if the list corresponding to the result of the looked-up fragment ID does not exist in the fragment look-up table).

Consider **claim 3**, and **as it applies to claim 1 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose the claimed method, wherein the step of transmitting the received packet includes attaching the index as the tag to a packet that has been previously received and stored in a fragment buffer and transmitting the previously received and stored packet to the corresponding one of the protocol processors, if the received packet is the first fragment and the list corresponding to the result of the looked-up fragment exists in the fragment look-up table (in Varma et al. reference, paragraph 0032, lines 13-20 which disclose that a determination is made as to whether the packet count value is zero. If the count value is not zero, the queue already exists, thus disclosing an existing queue for the received packet); (in Crow et al. reference, Fig. 3 that shows a sample address translation entry for a first fragment; Flowchart of Fig. 4, blocks 110-114; paragraphs 0039-0041 that disclose the process of generating an index for a primary fragment (fragment-context 92 shown in Fig. 3) for the identified translation entry 90, so that the primary and the related secondary fragments may be translated and transmitted to their destination host/port); and (in Rana et al. reference, Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a fragment from a known session (i.e. belonging to an existing list as disclosed in paragraphs 0020-0023), a fragment id is associated with the data packet, thus disclosing a method wherein the step of transmitting the received packet includes attaching the index as the tag to a packet that has been previously received and stored in a fragment buffer and transmitting the previously received and stored packet to the

corresponding one of the protocol processors, if the received packet is the first fragment and the list corresponding to the result of the looked-up fragment exists in the fragment look-up table).

Consider **claim 4**, and **as it applies to claim 1 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose the claimed method, wherein if the received packet is not the first fragment (in Crow et al. reference, Flowchart of Fig. 4, blocks 108, 118, 120; paragraph 0042, lines 1-4 that disclose the processing of a received packet that is not the first fragment), further comprising the steps of:

looking-up the fragment ID of the received packet and comparing the result of the looked-up fragment ID with each list of the fragment look-up table, to determine if there is a corresponding list; entering the result of the fragment ID looked-up for the received packet into the fragment look-up table, if the list corresponding to the result of the looked-up fragment does not exist in the fragment look-up table; and storing the received packet in a fragment buffer (in Varma et al. reference, paragraph 0032, lines 6-13 which disclose that a determination is made as to whether the received packet is a first block of data associated with this queue, that is if the queue was empty upon the arrival of the packet; this is achieved by checking packet count value for zero. If the count value is zero, the head pointer and the tail pointer are both set to the address of an allocated block in the data memory to store the received packet, as this is the only packet associated with the queue, thus disclosing a method wherein if the received

packet is not the first fragment packet, looking-up a fragment ID of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list, and entering the result of the fragment ID looked-up for the received packet into the fragment look-up table, if the list corresponding to the result of the looked-up fragment does not exist in the fragment look-up table; and storing the received packet in a fragment buffer);

Consider **claim 5**, and **as it applies to claim 4 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., show and disclose a method of the claimed invention, wherein if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table (in Varma et al. reference, paragraph 0032, lines 13-20 which disclose that a determination is made as to whether the packet count value is zero. If the count value is not zero, the queue already exists, thus disclosing an existing queue for the received packet), further comprising the steps of: determining whether the index corresponding to the result of the tunnel ID look-up exists in the corresponding list (in Ganesan et al. reference, Flowchart of Fig. 11, blocks 1120 and 1122; paragraph 0177 which discloses that for received packets, based on the tunnel ID of the packet, NAT (Network Address Translation) lookups and mappings are applied, thereby disclosing looking up a tunnel ID of the received packet from a tunnel ID look-up table); and

attaching the index as the tag to the received packet and transmitting the received packet to the corresponding one of the protocol processors, if the index exists in the corresponding list (in Crow et al. reference, Fig. 4, block 116, paragraph 0039 that discloses the translation and subsequent transmission process for the primary fragment; Flowchart of Fig. 4, block 124; paragraph 0043, lines 7-13 which disclose that the secondary fragment is translated using the identified entry 90 and transmitted to one of the protocol processors); and (in Rana et al. reference, Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a fragment from a known session (i.e. belonging to an existing list as disclosed in paragraphs 0020-0023), a fragment id is associated with the data packet, thus if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table, so as to be able to attach the index as a tag to the received packet and transmit the received packet to the corresponding one of the protocol processors, if the index exists in the corresponding list).

Consider **claim 6**, and **as it applies to claim 5 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose the claimed method, comprising the step of storing the received packet in the fragment buffer, if the index does not exist in the corresponding list (in Varma et al. reference, paragraph 0032, lines 13-20 which disclose that a determination is made as to whether the packet count value is zero. If the count value is not zero, the queue already exists; the value of

the current tail link for the queue is modified to point to a newly allocated block in data memory to store the received packet); and (in Crow et al. reference, Flowchart of Fig. 4, blocks 108 and 120; paragraph 0042, lines 14-17 which disclose that the secondary fragment 34 is stored in the fragment memory 84, if a fragment-context 92 does not exist for the secondary fragment 34 in the translation table 82).

Consider **claim 7**, Crow et al. show and disclose an apparatus for distributing a plurality of packets to a plurality of protocol processors (Fig.1, that shows a Border Router 16 receiving a plurality of packets from the Internet 22, and distributing the packets to a plurality of protocol processors 24; paragraph 0018 discloses the same details) comprising:

a dependant interface for transmitting the packet attached with the index to the corresponding one of the protocol processors (In Fig. 1, shown by the left link 20 between the Border Router 16 and Protocol Processor Hosts 24; Fig. 3 that shows a fragment context (an index) attached to the translated packet entry for transmission to its destination host/port).

However, Crow et al. do not explicitly disclose an apparatus comprising a receiving unit for receiving the packets from a network; a fragment look-up table storage unit for storing fragment look-up table into which the result of a fragment looked-up on the received packet is entered; a fragment look-up device for comparing the result of the fragment looked-up on the received packet with each list of the fragment look-up table, to determine whether the list corresponding to the result exists; a tunnel ID look-up table

storage unit for storing a tunnel ID look-up table having lists of indexes indicating the protocol processors corresponding to the tunnel IDs of the packets, respectively; and a tunnel ID look-up device for searching the index corresponding to the result of the tunnel ID looked-up on the received packet from the tunnel ID look-up table to attach the index as a tag to the received packet***; wherein, when the list of the fragment look-up table is not accessed during a predetermined time, the list is removed from the fragment look-up table.***

In the same field of endeavor, Rana et al. disclose an apparatus comprising a receiving unit for receiving the packets from a network (Fig. 1, Input 12 and Input Interface 14 forming a receiving unit for receiving the packets from a network; paragraph 0019 that discloses the same details);

***wherein, when the list of the fragment look-up table is not accessed during a predetermined time, the list is removed from the fragment look-up table*** **(paragraph 0027 which discloses that fragment reassembly unit 28 (shown in Fig. 1) must also check for time out conditions, to ensure that all of the datagram fragments associated with a particular PDU are received, otherwise the full PDU cannot be reassembled and both the resources of fragment memory 30 and link list memory 24 would eventually fill with these unresolved fragments and be rendered useless; further disclosing that to prevent this situation, a timeout condition is generated after a fragment or fragments have been in the fragment reassembly unit for more than a programmable, predetermined amount of time;**

**and once the timeout condition has occurred, all of the fragments for that PDU are discarded and the links in the link list memory 24 are deallocated).**

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide an apparatus comprising a receiving unit for receiving the packets from a network, wherein, when the list of the fragment look-up table is not accessed during a predetermined time, the list is removed from the fragment look-up table, as taught by Rana et al., in the apparatus of Crow et al., so that the input packets can be received by the apparatus, and to free up the memory resources that hold message fragments when all the fragments for a particular message are not received within a predetermined time.

However, Crow et al., as modified by Rana et al., do not explicitly disclose an apparatus comprising a fragment look-up table storage unit for storing fragment look-up table into which the result of a fragment looked-up on the received packet is entered; a fragment look-up device for comparing the result of the fragment looked-up on the received packet with each list of the fragment look-up table, to determine whether the list corresponding to the result exists; a tunnel ID look-up table storage unit for storing a tunnel ID look-up table having lists of indexes indicating the protocol processors corresponding to the tunnel IDs of the packets, respectively; and a tunnel ID look-up device for searching the index corresponding to the result of the tunnel ID looked-up on the received packet from the tunnel ID look-up table to attach the index as a tag to the received packet.

In the same field of endeavor, Varma et al. disclose an apparatus comprising a fragment look-up table storage unit for storing fragment look-up table into which the result of a fragment looked-up on the received packet is entered (Fig. 3, Control Memory 200 used for storing fragment look-up table with a plurality of lists within it, each list represented by a head pointer, a tail pointer and a packet count value, the lists storing fragment id (queue id) value of received fragmented packets stored in the data memory 210 and referenced by the link memory 220); and a fragment look-up device for comparing the result of the fragment looked-up on the received packet with each list of the fragment look-up table, to determine whether the list corresponding to the result exists (paragraph 0032 which discloses a queue processor that creates a new list and stores the received packet data each time a first new packet is received and for every subsequent packet received, compares the queue id of the incoming packet with the queues already in the control memory to place the incoming packet in the existing queue).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide an apparatus comprising a fragment look-up table storage unit for storing fragment look-up table into which the result of a fragment looked-up on the received packet is entered; a fragment look-up device for comparing the result of the fragment looked-up on the received packet with each list of the fragment look-up table, to determine whether the list corresponding to the result exists, as taught by Varma et al., in the apparatus of Crow et al., as modified by Rana et al., so

that the related packets can be grouped in a common queue (list) for subsequent translation and transmission to their destination host/port.

However, Crow et al., as modified by Rana et al. and Varma et al., do not explicitly disclose an apparatus comprising a tunnel ID look-up table storage unit for storing a tunnel ID look-up table having lists of indexes indicating the protocol processors corresponding to the tunnel IDs of the packets, respectively; and a tunnel ID look-up device for searching the index corresponding to the result of the tunnel ID looked-up on the received packet from the tunnel ID look-up table to attach the index as a tag to the received packet.

In the same field of endeavor, Ganesan et al. disclose an apparatus comprising a tunnel ID look-up table storage unit for storing a tunnel ID look-up table having lists of indexes indicating the protocol processors corresponding to the tunnel IDs of the packets, respectively (Figure 6B, remote hash table rhashtbl_t, that includes storage for vpn_id which corresponds to tunnel information table); and
a tunnel ID look-up device for searching the index corresponding to the result of the tunnel ID looked-up on the received packet from the tunnel ID look-up table to attach the index as a tag to the received packet (Fig. 8, VPN/IKE Module 830 performing the function of a tunnel ID look-up device; Flowchart of Fig. 11, blocks 1120 and 1122; paragraph 0177 which discloses that for received packets, based on the tunnel ID of the packet, NAT (Network Address Translation) lookups and mappings are applied, thereby disclosing a tunnel ID look-up device for searching the index corresponding to the result

of the tunnel ID looked-up on the received packet from the tunnel ID look-up table to attach the index as a tag to the received packet).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide an apparatus containing a tunnel ID look-up table storage unit for storing a tunnel ID look-up table having lists of indexes indicating the protocol processors corresponding to the tunnel IDs of the packets, respectively, and a tunnel ID look-up device for searching the index corresponding to the result of the tunnel ID looked-up on the received packet from the tunnel ID look-up table to attach the index as a tag to the received packet, as taught by Ganesan et al., in the apparatus of Crow et al. as modified by Rana et al. and Varma et al., so that encapsulated received packets can be securely delivered through the firewall of the receiving node.

Consider **claim 8**, and **as it applies to claim 7 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose an apparatus, wherein if the list corresponding to the result of the looked-up fragment does not exist in the fragment look-up table, the fragment look-up device newly enters the result of the looked-up fragment and the index into the fragment look-up table, if the received packet is a first fragment (in Crow et al. reference, Flowchart of Fig. 4, blocks 108-116; paragraph 0039 that discloses the processing of a received packet that is the first fragment); (in Varma et al. reference, paragraph 0032, lines 6-13 which disclose that a determination is made as to whether the received packet is a first block of data associated with this queue, that is if the queue was empty upon the arrival of the data;

by checking the packet count value for zero. If the count value is zero, the head pointer and the tail pointer are both set to the address of an allocated block in the data memory to store the received packet, as this is the only packet associated with the queue, thereby disclosing creating a new list with the fragment id (queue id), if the list corresponding to the result of the looked-up fragment ID does not exist in the fragment look-up table); (in Crow et al. reference, Fig. 3 that shows a sample address translation entry; Flowchart of Fig. 4, blocks 110-114; paragraph 0040 that discloses the process of generating an index (fragment-context 92 shown in Fig. 3) for the identified translation entry 90, thereby disclosing generating an index); and (in Rana et al. reference, Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a new fragment, in which case a fragment id (index) is assigned to the data packet and stored in the link list); thus disclosing an apparatus wherein if the list corresponding to the result of the looked-up fragment does not exist in the fragment look-up table, the fragment look-up device newly enters the result of the looked-up fragment and the index into the fragment look-up table, if the received packet is a first fragment); and

newly enters the result of the looked-up fragment into the fragment look-up table, if the received packet is not the first fragment (in Crow et al. reference, Flowchart of Fig. 4, blocks 108, 118, 120; paragraph 0042, lines 1-4 that disclose the processing of a received packet that is not the first fragment); (in Varma et al. reference, paragraph 0032, lines 6-13 which disclose that a determination is made as to whether the received packet is a first block of data associated with this queue, that is if the queue was empty

upon the arrival of the packet; this is achieved by checking packet count value for zero. If the count value is zero, the head pointer and the tail pointer are both set to the address of an allocated block in the data memory to store the received packet, as this is the only packet associated with the queue, thus disclosing an apparatus wherein if the list corresponding to the result of the looked-up fragment does not exist in the fragment look-up table, the fragment look-up device newly enters the result of the looked-up fragment into the fragment look-up table, if the received packet is not the first fragment).

Consider **claim 9**, and **as it applies to claim 7 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose an apparatus comprising, if the list corresponding to the result of the looked-up fragment and including the index does not exist in the fragment look-up table, a fragment buffer for storing the received packet if the received packet is not the first fragment (in Crow et al. reference, Flowchart of Fig. 4, blocks 108, 118, 120; paragraph 0042 that discloses the processing of a received packet that is not the first fragment); (in Varma et al. reference, paragraph 0032, lines 6-13 which disclose that a determination is made as to whether the received packet is a first block of data associated with this queue, that is if the queue was empty upon the arrival of the packet; this is achieved by checking packet count value for zero. If the count value is zero, the head pointer and the tail pointer are both set to the address of an allocated block in the data memory to store the received packet, as this is the only packet associated with the queue, thus disclosing an apparatus wherein if the list corresponding to the result of the looked-up fragment does

not exist in the fragment look-up table, the fragment look-up device stores the received packet in a fragment buffer, if the received packet is not the first fragment).

Consider **claim 10**, and **as it applies to claim 9 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose an apparatus wherein if the list corresponding to the result of the looked-up fragment and including the index exists in the fragment look-up table, the fragment look-up device attaches the index as the tag to the received packet to transmit the received packet to the corresponding one of the protocol processors (in Varma et al. reference, paragraph 0032, lines 13-20 which disclose that a determination is made as to whether the packet count value is zero. If the count value is not zero, the queue already exists, thus disclosing an existing queue for the received packet); (in Crow et al. reference, Fig. 3 that shows a sample address translation entry for a first fragment; Flowchart of Fig. 4, blocks 110-114; paragraphs 0039-0043 that disclose the process of generating an index for a primary fragment (fragment-context 92 shown in Fig. 3) for the identified translation entry 90, so that the primary and the related secondary fragments may be translated and transmitted to their destination host/port); and (in Rana et al. reference, Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a fragment from a known session (i.e. belonging to an existing list as disclosed in paragraphs 0020-0023), a fragment id is associated with the data packet, thus disclosing an apparatus wherein if the list corresponding to the result of the looked-up fragment and including the index exists in

the fragment look-up table, the fragment look-up device attaches the index as the tag to the received packet to transmit the received packet to the corresponding one of the protocol processors).

Consider **claim 11**, and **as it applies to claim 9 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose an apparatus wherein in the case of the received packet being the first fragment, the fragment look-up device attaches the index as the tag to each packet being a subsequent fragment following the first fragment and being stored in the fragment buffer to transmit each subsequent fragment packet via the dependant interface to the corresponding one of the protocol processors, if the list conforming to the result of the looked-up fragment exists in the fragment look-up table (in Varma et al. reference, paragraph 0032, lines 13-20 which disclose that a determination is made as to whether the packet count value is zero. If the count value is not zero, the queue already exists, thus disclosing an existing queue for the received packet); (in Crow et al. reference, Fig. 3 that shows a sample address translation entry for a first fragment; Flowchart of Fig. 4, blocks 110-114; paragraphs 0039-0041 that disclose the process of generating an index for a primary fragment (fragment-context 92 shown in Fig. 3) for the identified translation entry 90, so that the primary and the related secondary fragments may be translated and transmitted to their destination host/port); and (in Rana et al. reference, Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a fragment from a known session (i.e.

belonging to an existing list as disclosed in paragraphs 0020-0023), a fragment id is associated with the data packet, thus disclosing an apparatus wherein in the case of the received packet being the first fragment, the fragment look-up device attaches the index as the tag to each packet being a subsequent fragment following the first fragment and being stored in the fragment buffer to transmit each subsequent fragment packet via the dependant interface to the corresponding one of the protocol processors, if the list conforming to the result of the looked-up fragment exists in the fragment look-up table).

Consider **claim 14**, Crow et al. show and disclose ***a method for receiving a plurality of packets from a network and distributing the packets to a plurality of protocol processors*** **(Fig.1, that shows a Border Router 16 receiving a plurality of packets from the Internet 22, and distributing the packets to a plurality of protocol processors 24; paragraph 0018 discloses the same details)** ***comprising the steps of:***

***if a received packet is a fragmented packet, determining whether the received packet is a first fragment packet*** **(Flowchart of Fig. 4, blocks 102 and 108 that test for a fragmented received packet and then check if it is the first (primary) fragment packet; paragraph 0037, lines 1-5 that disclose testing for a fragmented packet; paragraph 0038 that discloses a test for the fragmented received packet being the first such packet);**

***searching an index indicating one of the protocol processors*** **(Fig. 1, Translation Table 82 with entries in it; Fig. 3 that shows an entry matching the IP and**

**Transport Header information (shown in Fig. 2A) in the primary fragment, after searching the translation table 82 for a matching protocol processor);**
***entering the index into the corresponding list of the fragment look-up table*** **(Fig. 3, a group of Fragment Contexts shown below Fragment Context 92 and representing a list, for the secondary segments that were previously stored, being updated with the fragment context of the primary fragment; Fig. 4, blocks 114-116; paragraphs 0039-0040 that describe generating a fragment-context 92 (shown in Fig. 3) for the identified translation entry and applying it to the secondary fragments);** ***and***
***attaching the index as a tag to the received packet and transmitting the received packet to the corresponding one of the protocol processors*** **(Fig. 4, block 116, paragraph 0039 that discloses the translation and subsequent transmission process for the primary fragment).**

However, Crow et al. do not disclose ***a method wherein if the received packet is the first fragment packet, looking up a tunnel ID of the received packet from a tunnel ID look-up table; looking-up a fragment ID of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list; and if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table, and wherein the other received fragment packets are stored in a***

***fragment buffer, wherein a list is stored in the fragment look-up table for each of the fragmented packets, wherein, if the received fragment packet is determined to be the first fragment packet, searching the look-up table for a first list that corresponds to the other received fragment packets, wherein the other received fragment packets together with the first fragment packet form a datagram, and wherein if the received fragment packet is determined to be the first fragment packet and the first list is found in the look-up table, editing the list to update the index to be valid and searching the fragment buffer for the other received fragment packets and transmitting the found other received fragments based on the updated valid index of the first list without assembling the fragment packets to form the datagram; and*** wherein, if the received fragment packet is determined to be one of the other fragment packets, searching the look-up table for the first list, and if the first list is not present, generating the first list comprising source address, destination address and an index and storing the one of the other fragment packets in the fragment buffer.

In the same field of endeavor, Ganesan et al. disclose ***a method for looking up a tunnel ID of the received packet from a tunnel ID look-up table*** **(Flowchart of Fig. 11, blocks 1120 and 1122; paragraph 0177 which discloses that for received packets, based on the tunnel ID of the packet, NAT (Network Address Translation) lookups and mappings are applied, thereby disclosing looking up a tunnel ID of the received packet from a tunnel ID look-up table).**

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide a method for looking up a tunnel ID of the received packet from a tunnel ID look-up table, as taught by Ganesan et al., in the method of Crow et al., so that encapsulated received packets can be securely delivered through the firewall of the receiving node.

However, Crow et al., as modified by Ganesan et al., do not specifically disclose ***a method wherein if the received packet is the first fragment packet, looking-up a fragment ID of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list; and if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table, and wherein the other received fragment packets are stored in a fragment buffer, wherein a list is stored in the fragment look-up table for each of the fragmented packets, wherein, if the received fragment packet is determined to be the first fragment packet, searching the look-up table for a first list that corresponds to the other received fragment packets, wherein the other received fragment packets together with the first fragment packet form a datagram, and wherein if the received fragment packet is determined to be the first fragment packet and the first list is found in the look-up table, editing the list to update the index to be valid and searching the fragment buffer for the other received fragment packets and transmitting the found other***

***received fragments based on the updated valid index of the first list without assembling the fragment packets to form the datagram; and*** wherein, if the received fragment packet is determined to be one of the other fragment packets, searching the look-up table for the first list, and if the first list is not present, generating the first list comprising source address, destination address and an index and storing the one of the other fragment packets in the fragment buffer.

In the same field of endeavor, Varma et al. disclose ***a method wherein if the received packet is the first fragment packet, looking-up a fragment ID of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list*** **(Fig. 3; that shows Data Memory 210 for storing packets, Control Memory 200 for storing the fragment look-up table with head and tail pointers and other data (including packet count) for each list, and Link Memory 220 with pointers to keep track of the related packets of the list in the data memory; paragraph 0032, lines 6-13 which disclose that a determination is made as to whether the received packet is a first block of data associated with this queue, that is if the queue was empty upon the arrival of the packet; by checking the packet count value for zero. If the packet count value is zero, the head pointer and the tail pointer are both set to the address of an allocated block in the data memory to store the received packet, as this is the only packet associated with the queue, thus disclosing a method wherein if the received packet is the first fragment packet, looking-up a**

**fragment ID (queue id) of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list); *and***

***wherein the other received fragment packets are stored in a fragment buffer* (Fig. 3 that shows Data Memory 210 (fragment buffer) where fragment packets received out-of-order are stored; paragraph 0030 discloses the same details),**

***wherein a list is stored in the fragment look-up table for each of the fragmented packets* (Fig. 3 that further shows a link memory 220 (fragment look-up table) that stores a list of pointers that locate different fragments or packets of a message stored in the Data Memory 210; paragraph 0030 discloses the same details),**

***wherein, if the received fragment packet is determined to be the first fragment packet* (Fig. 3 that also shows a Control Memory 200, each row of which corresponds to a single or multi-packet message stored in the Data Memory 210, the row including the head and tail pointer of the queue of packets for a message and other fields for storing packet count and status of the queue processing, the head pointer pointing to the first received packet, which will be the first packet of the message when all the packets have been received and stored; paragraphs 0030-0031 disclose the details shown),**

***searching the look-up table for a first list that corresponds to the other received fragment packets* (Fig. 3 which shows that starting with a message ID (that corresponds to the row in the Control Memory), the head pointer (set to 21) is**

**used to index in the Data Memory (at row 21) to locate the content of the first packet (e.g. packet 1), and also used to index in the Link Memory to find the pointer (34) for the next packet (e.g. packet 2); this next packet pointer can then be similarly used to locate the packet data for the next packet (e.g. packet 2) in the Data Memory (at row 34) and a pointer (67) to a subsequent packet (e.g. packet 3) in the Link Memory at row 34, repeating the process until a null (empty) value for a pointer is reached in the Link Memory, indicating the last packet; paragraphs 0030-0031 describe the functioning of the linked-list disclosed and shown in Fig. 3),**

***wherein the other received fragment packets together with the first fragment packet form a datagram*** **(paragraph 0030 which discloses that if more than one block is required to store the data for a particular queue, the data is stored in a plurality of blocks)*, and***

***wherein if the received fragment packet is determined to be the first fragment packet and the first list is found in the look-up table, editing the list to update the index to be valid and searching the fragment buffer for the other received fragment packets and transmitting the found other received fragments based on the updated valid index of the first list without assembling the fragment packets to form the datagram*** **(this claimed feature is an inherent design detail of a linked-list, wherein as the packets in a message are received, they are stored in the next available row in the Data Memory (i.e. in non-contiguous blocks), and the pointer values in the corresponding rows in the Link Memory are updated; after all the**

**packets have been received, the packets stored in the Data Memory may be transmitted without reassembling them again by simply tracing their pointers, starting with the head pointer for the message stored in the Control Memory).**

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide a method wherein if the received packet is the first fragment packet, looking-up a fragment ID of the received packet, and comparing the result of the looked-up fragment ID with each list of a fragment look-up table into which the results of fragment looked-ups for other received packets are entered, to determine if there is a corresponding list, and wherein the other received fragment packets are stored in a fragment buffer, wherein a list is stored in the fragment look-up table for each of the fragmented packets, wherein, if the received fragment packet is determined to be the first fragment packet, searching the look-up table for a first list that corresponds to the other received fragment packets, wherein the other received fragment packets together with the first fragment packet form a datagram, and wherein if the received fragment packet is determined to be the first fragment packet and the first list is found in the look-up table, editing the list to update the index to be valid and searching the fragment buffer for the other received fragment packets and transmitting the found other received fragments based on the updated valid index of the first list without assembling the fragment packets to form the datagram, as taught by Varma et al., in the method of Crow et al., as modified by Ganesan et al., so that the related packets received in the out of order sequence, may be organized in a single list for subsequent translation and transmission to a common destination host/port.

However, Crow et al., as modified by Ganesan et al. and Varma et al., do not specifically disclose ***a method wherein if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table;*** and wherein, if the received fragment packet is determined to be one of the other fragment packets, searching the look-up table for the first list, and if the first list is not present, generating the first list comprising source address, destination address and an index and storing the one of the other fragment packets in the fragment buffer. Although Crow et al. do disclose creating a fragment context for a primary fragment and associating it with the related secondary fragments for subsequent translation and transmission of the related secondary fragments, Crow et al. do not specifically associate it with a list in the fragment look-up table.

In the same field of endeavor, Rana et al. disclose ***a method wherein if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table*** **(Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a fragment from a known session (i.e. belonging to an existing link list, as discussed in paragraphs 0020-0023), a fragment id is associated with the data packet, thus disclosing a method wherein if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table);** and

wherein, if the received fragment packet is determined to be one of the other fragment packets, searching the look-up table for the first list, and if the first list is not present, generating the first list comprising source address, destination address and an index and storing the one of the other fragment packets in the fragment buffer (Fig. 3c that shows the data structure for a link list memory comprising a session id (SID) field; paragraph 0037 which discloses that the link list memory 24 (shown in Fig. 1) is used to associate the blocks in packet memory 20 that form the PDUs (Packet Data Units) from the same session or traffic flow; paragraph 0020 which discloses that the link lists in the link list memory 24 are used by the queue engine 10 to track pointers associated with data packets stored in packet memory 20; paragraphs 0021-0023 which further disclose that a session could be identified and assigned a session id based upon the source address, destination address and any other field or combination of fields from the header of the data packet which form a unique identifier (corresponding to the first list); further disclosing that if the data packet is a fragment from a known session (existing first list), a fragment id is associated with the data packet, or if the data packet is a new fragment (non-existing first list), a fragment id based on the session id is assigned to the data packet).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to provide a method wherein if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table, entering the index into the corresponding list of the fragment look-up table, and wherein, if the received fragment packet is determined to be one of the other fragment packets,

searching the look-up table for the first list, and if the first list is not present, generating the first list comprising source address, destination address and an index and storing the one of the other fragment packets in the fragment buffer, as taught by Rana et al., in the method of Crow et al., as modified by Ganesan et al. and Varma et al., so that the related packets received in the out of order sequence, may be organized in a single list for subsequent translation and transmission to a common destination host/port.

Consider **claim 15**, and **as it applies to claim 14 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further disclose the claimed method, wherein the fragment look-up table further comprises a field indicating storage location of respective at least one other fragment packet in the fragment buffer (in Rana et al. reference, Fig. 3c, field marked "Next" that indicates storage location of respective at least one other fragment packet in the fragment buffer; paragraph 0037 that discloses the same details).

Consider **claim 16**, and **as it applies to claim 1 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further show and disclose a method wherein, if the received packet is the first fragment packet (in Crow et al. reference, Flowchart of Fig. 4, blocks 108-116; paragraph 0039 that discloses the processing of a received packet that is the first fragment),
searching an index indicating one of the protocol processors (in Crow et al. reference, Fig. 3, that shows a searched Address Translation Entry field 90 in the Address

Translation Table 82 of Fig. 1, that matches the protocol processor and destination port in the IP and Transport Headers of the first fragment shown in Fig. 2A); and

corresponding to the tunnel ID of the received packet from a tunnel ID look-up table (in Ganesan et al. reference, Flowchart of Fig. 11, blocks 1120 and 1122; paragraph 0177 which discloses that for received packets, based on the tunnel ID of the packet, NAT (Network Address Translation) lookups and mappings are applied, thereby disclosing looking up a tunnel ID of the received packet from a tunnel ID look-up table), and

if the list corresponding to the result of the looked-up fragment ID exists in the fragment look-up table (in Varma et al. reference, paragraph 0032, lines 13-20 which disclose that a determination is made as to whether the packet count value is zero. If the count value is not zero, the queue already exists, thus disclosing an existing queue for the received packet),

updating the index into the corresponding list of the fragment look-up table (in Rana et al. reference, Fig. 1; paragraph 0023 that discloses analyzing packet header to determine whether a data packet is a fragment; and if the data packet is a new fragment, in which case a fragment id (index) is assigned to the data packet and stored in the link list); and

transmitting the other received fragment packets stored in a fragment buffer based on the updated index stored in the corresponding list of the fragment look up table (in Crow et al. reference, Fig. 3, Fragment Context field 92, an index updated in the Address Translation Entry field 90 after the first fragment packet is received; paragraphs 0039-0041 that disclose generating a fragment-context 92 for the identified translation entry

90 and associating the corresponding fragment-contexts of the secondary entries with that of the first fragment packet, so that the other (secondary) fragment packets received prior to the first fragment packet can be appropriately translated and transmitted to their destination as well).

Consider **claim 18**, and **as it applies to claim 1 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further disclose the claimed method, wherein the other received fragment packets are stored in a fragment buffer, wherein a list is stored in the fragment look-up table for each of the fragmented packets, wherein the fragment look-up table is stored separately from the fragment memory, wherein, if the received fragment packet is determined to be the first fragment packet, searching the look-up table for a first list that corresponds to the other received fragment packets, wherein the other received fragment packets together with the first fragment packet form a datagram (in Varma et al. reference, Fig. 3, Data Memory 210 that stores the other received fragment packets (received out of order and stored in the order received, intermixed with fragments of other received messages); Fig. 3, Link Memory 220 equivalent to fragment look-up table, that is used to store pointers (to addresses in the Data Memory) to the stored fragments; Fig. 3 clearly showing that the Data Memory 210 (fragment memory) is stored separately from the Link Memory 220 (fragment look-up table); paragraphs 0030-0031 describe the workings of the linked lists composed of the Control Memory 200, Data Memory 210, and Link Memory 220; paragraph 0032 further teaches handling of fragments of a packet received out of sequence, i.e. receiving the

first fragment after later fragments have already been received and saved by searching the Control Memory 200 wherein the fragment count value is zero (empty queue) or count > zero (some fragments have already been saved); paragraph 0030 which also discloses that if more than one block is required to store the data for a particular queue, the data is stored in a plurality of blocks, thereby disclosing that the other received fragment packets together with the first fragment packet form a datagram).

Consider **claim 19**, and **as it applies to claim 18 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further disclose the claimed method, wherein, if the received fragment packet is determined to be the first fragment packet (in Varma et al. reference, Fig. 3 that shows a Control Memory 200, each row of which corresponds to a single or multi-packet message stored in the Data Memory 210, the row including the head and tail pointer of the queue of packets for a message and other fields for storing packet count and status of the queue processing, the head pointer pointing to the first received packet, which will be the first fragment when all the fragments have been received and stored; paragraphs 0030-0031 disclose the details shown), and

the first list is found in the look-up table, editing the list to update the index to be valid and searching the fragment buffer for the other received fragment packets and transmitting the found other received fragments based on the updated valid index of the first list without assembling the fragment packets to form the datagram (this claimed feature is an inherent design detail of a linked-list, wherein as the packets in a message

are received, they are stored in the next available row in the Data Memory (i.e. in non-contiguous blocks), and the pointer values in the corresponding rows in the Link Memory are updated; after all the packets have been received, the packets stored in the Data Memory may be transmitted without reassembling them again by simply tracing their pointers, starting with the head pointer for the message stored in the Control Memory).

**Claim 17** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Crow et al. (U.S. Patent Application Publication # 2002/0161915 A1)** in view of **Ganesan et al. (U.S. Patent Application Publication # 2003/0069973 A1)** and further in view of **Varma et al. (U.S. Patent Application Publication # 2004/0037302 A1)** and further in view of **Rana et al. (U.S. Patent Application Publication # 2002/0095512 A1)** and further in view of **Hui et al. (U.S. Patent Application Publication # 2004/0151197 A1).**

Consider **claim 17**, and **as it applies to claim 14 above**, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., further disclose the claimed method, wherein if the first fragment packet is not received, the index in the first list that corresponds to the other received fragment packets is set to invalid (in Varma et al. reference, Fig. 3, field marked “Other Fields” in each row of Control Memory 200; paragraph 0031 which discloses that the other fields may include a count of packets, and status of the processing for the queue).

However, Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., do not specifically disclose that the index is set to invalid.

In the same field of endeavor, Hui discloses the claimed method, including setting the index to invalid (paragraph 0017 which discloses that the queuing memory has row that each store queuing elements for only one output port, and may include a pointer to a linked list of other queuing elements for the flow; further disclosing that each queuing element may include a valid flag which is set to valid for a packet in the queue and changed to invalid after the queuing element is dequeued).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to set the index in the first list that corresponds to the other received fragment packets to invalid, if the first fragment packet is not received, as taught by Hui, in the method of Crow et al., as modified by Ganesan et al., Varma et al., and Rana et al., so that messages received with dropped packets may be flushed from the buffer memory.

### ***Response to Arguments***

Applicant's arguments filed 04/16/2009 have been fully considered but they are not persuasive. The examiner has concluded that the cited prior art teach and fully disclose each and every claim element. Claims 1-11 and 14-19 are considered obvious over the cited prior art and not novel, and therefore not allowable. The examiner's response to the applicant's arguments is shown below:

Consider **claim 1**. The applicant keeps repeating that the examiner's position is not understood (Remarks page 13). The examiner has made every effort to explain how linked-lists are constructed and how they function in the previous office actions. If the applicant cannot understand the examiner's position, the examiner would like to suggest reading the design and operational features of linked-lists in a computer science book.

The applicants further argue that Varma et al. do not disclose or suggest "searching if a corresponding list exists when the first fragment is received", further stating that in Varma, it is clear that when the data is received for the empty queue, no searches are performed. That is, if a queue is empty, there is no need to search for an existing list. The examiner begs to differ with this argument. In Varma's Control Memory 200 shown in Fig. 3, there may be several queues operating simultaneously (one for each row in table 200). All the rows are initially set to zero. When a fragment of a new message is received, the "Head, Tail, and Other Fields" of each row of Table 200 are searched to find a row (e.g. row 15) that is not already allocated to another message, so as not to overwrite that row. When a row with zero value in these fields is found, the first fragment is stored in the next available row (e.g. row 21) in the Data Memory 210, and the index of the Data Memory row (e.g. 21) is stored as the head pointer 21 of the message fragment list in the Control Memory at row 15. The Tail pointer and row 21 of the Link Table 220 are both set to zero to indicate no tail (i.e. only one fragment so far for this message). These fields get updated as other fragments arrive (out of order) and are stored at the next empty row in the Data Memory. The zero

values of the Tail pointer and the Link Memory are also updated. The examiner hopes that the process explained above clearly establishes that in Varma et al. reference, the search is needed, even for an empty queue.

The applicant further argues that there is no disclosure or even remote suggestion that the first fragment is received after the other packet fragments. The examiner disagrees with this assertion. The linked-lists are exclusively used for such processing, where transactions arrive in out of order sequence, and yet need not be sorted. The applicant keeps repeating the same argument about Varma et al. reference without analyzing how linked-lists operate. Yet, the examiner has made every effort to explain the concept, which should suffice.

The applicant's argument that Varma et al. reference does not disclose fragment IDs (or message IDs) requires no response, because these fields are a part of every message fragment, without which no one can reassemble the fragmented messages.

The applicant also argues that Rana et al. reference does not disclose "entering an index into the corresponding list". The examiner had cited the use of fragment id as an index in paragraph 0023 in support of this claim element in the previous office action and would like, as a response, to suggest the applicant review the cited paragraph one more time. As to the applicant's argument that Rana et al. do not disclose "the index of subsequent packets being invalid prior to receipt of the first packet", the examiner has found no such requirement in the text of claim 1.

No new arguments are presented for independent claims 7 and 14 and their dependent claims 2-6, 8-11, and 15-16, which therefore require no new response.

Consider **claim 17**. The applicant argues that the added reference of Hui et al. do not teach the claim elements already taught by other references cited for the independent claim 1. In a 103(a) rejection, each reference is not required to teach each and every claim feature. The examiner included Hui et al. reference to teach only the claim element of "setting an index to invalid", which it does.

Therefore, none of the claims 1-11 and 14-19 are allowable.

## *Conclusion*

**THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any response to this Office Action should be **faxed to** (571) 273-8300 **or mailed to**:

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Art Unit: 2443

**Hand-delivered responses** should be brought to

Customer Service Window
Randolph Building
401 Dulany Street
Alexandria, VA 22314

Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Kishin G. Belani whose telephone number is (571) 270-1768. The Examiner can normally be reached on Monday-Friday from 6:00 am to 5:00 pm.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Tonia Dollinger can be reached on (571) 272-4170. The fax phone number for the organization where this application or proceeding is assigned is (571) 273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free) or 703-305-3028.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist/customer service whose telephone number is (571) 272-0800.

*/K. G. B./*

*Examiner, Art Unit 2443*

July 21, 2009

/George C Neurauter, Jr./
Primary Examiner, Art Unit 2443